Figure 1

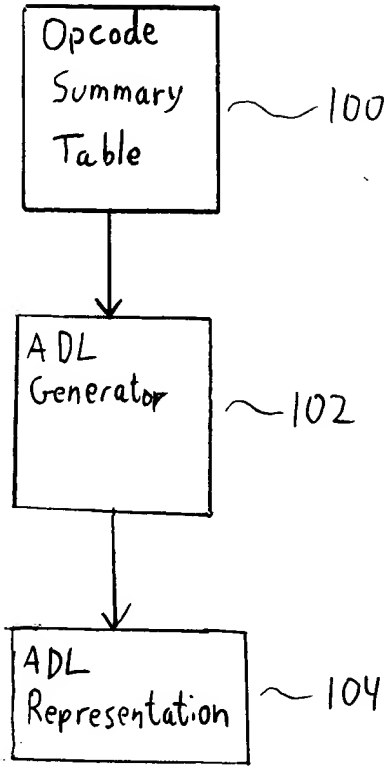

Opcode Summary Table
100
ADL Generator
102
ADL Representation
104

Figure 2

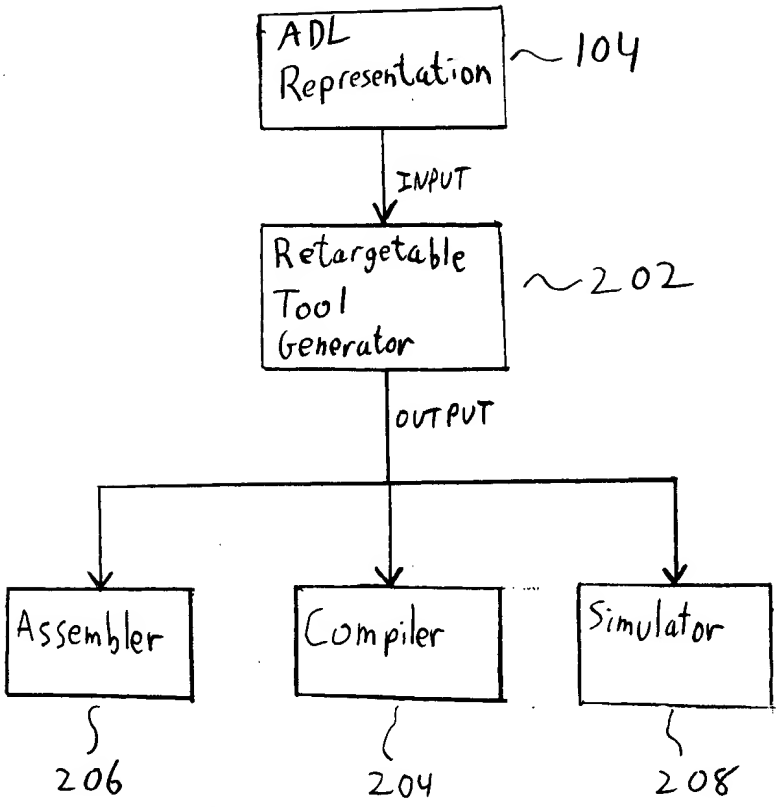

ADL Representation
104
INPUT
Retargetable Tool Generator
202
OUTPUT
Assembler
Compiler
Simulator
206
204
208

Figure 3

```
operation Ld_St_Imm_group
{ // The Load-Store Halfword Immediate Group
composition
     {
     // select one of the four the instructions.
     inst = (LDRH || LDRSB || LDRSH || STRH)
     // and include their operand fields.
     && cond && P && U && W && Rn && Rd
     && immed // immed is a joining of immedH and immedL.
     }
coding
     { // concatenate the opcode and operand fields
     self = cond:4 ## 0:3 ## P:1 ## U:1
     ## 1:1 ## W:1
     ## inst[2:2]:1 // the bit after the "W" operand.
     ## Rn ## Rd
     ## immed[4:7]:4 // immedH field.
     ## 1:1
     ## inst[1:0]:2 // the 2 bits between immedH & L.
     ## 1:1
     ## immed[0:3}:4 // immedL field.
     }
}
```

Figure 4

```
operation Ld_St_half_group
{
  composition
  {
    // select one of the four the instructions.
    grp = (Ld_St_Imm_group || Ld_St_Reg_group)
    // and include their operand fields.
    && cond && P && U && W && Rn && Rd
  }
  coding
  { // concatenate the opcode and operand fields
    self = cond:4 ## 0:3 ## P:1 ## U:1
           ## grp[11:11]:1 // 1st "*", the immed/reg indicator bit.
           ## W:1
           ## grp[10:10]:1 // 2nd "*", the first inst opcode bit.
           ## Rn ## Rd
           ## grp[9:6]:4 // 3rd "*", the immedH field, or 0's.
           ## 1:1
           ## grp[5:4]:2 // 4th "*", the other inst opcode bits.
           ## 1:1
           ## grp[3:0]:4 // 5th "*", the immedL or Rm field.
  }
                                    }
```

Figure 5

```
operation Ld_St_Imm_group
{
  composition
  {
    // select one of the four the instructions.
    inst = (LDRH || LDRSB || LDRSH || STRH)
    // and include their operand fields.
    && immed // immed is a joining of immedH and immedL.
  }
  coding
  { // blindly concatenate the opcode and operand fields
    self = 1:1 // imm indicator, opcode bit before "W" field.
           ## inst[2:2]:1 // the bit after the "W" operand.
           ## immed[4:7]:4 // immedH field.
           ## inst[1:0]:2 // the 2 bits between immedH & L.
           ## immed[0:3]:4 // immedL field.
  }
}
```

Figure 6

```
operation Ld_St_Reg_group
{
  composition
  {
    // select one of the four the instructions.
    inst = (LDRH || LDRSB || LDRSH || STRH)
    // and include their operand fields.
    && Rm // the extra register operand.
  }
  coding
  { // blindly concatenate the opcode and operand fields
    self = 0:1 // reg indicator, opcode bit before "W" field.
           ## inst[2:2]:1 // the bit after the "W" operand.
           ## 0:4 // equivalent to immedH field.
           ## inst[1:0]:2 // the 2 bits between immedH & L.
           ## Rm:4
  }
}
```

Figure 7

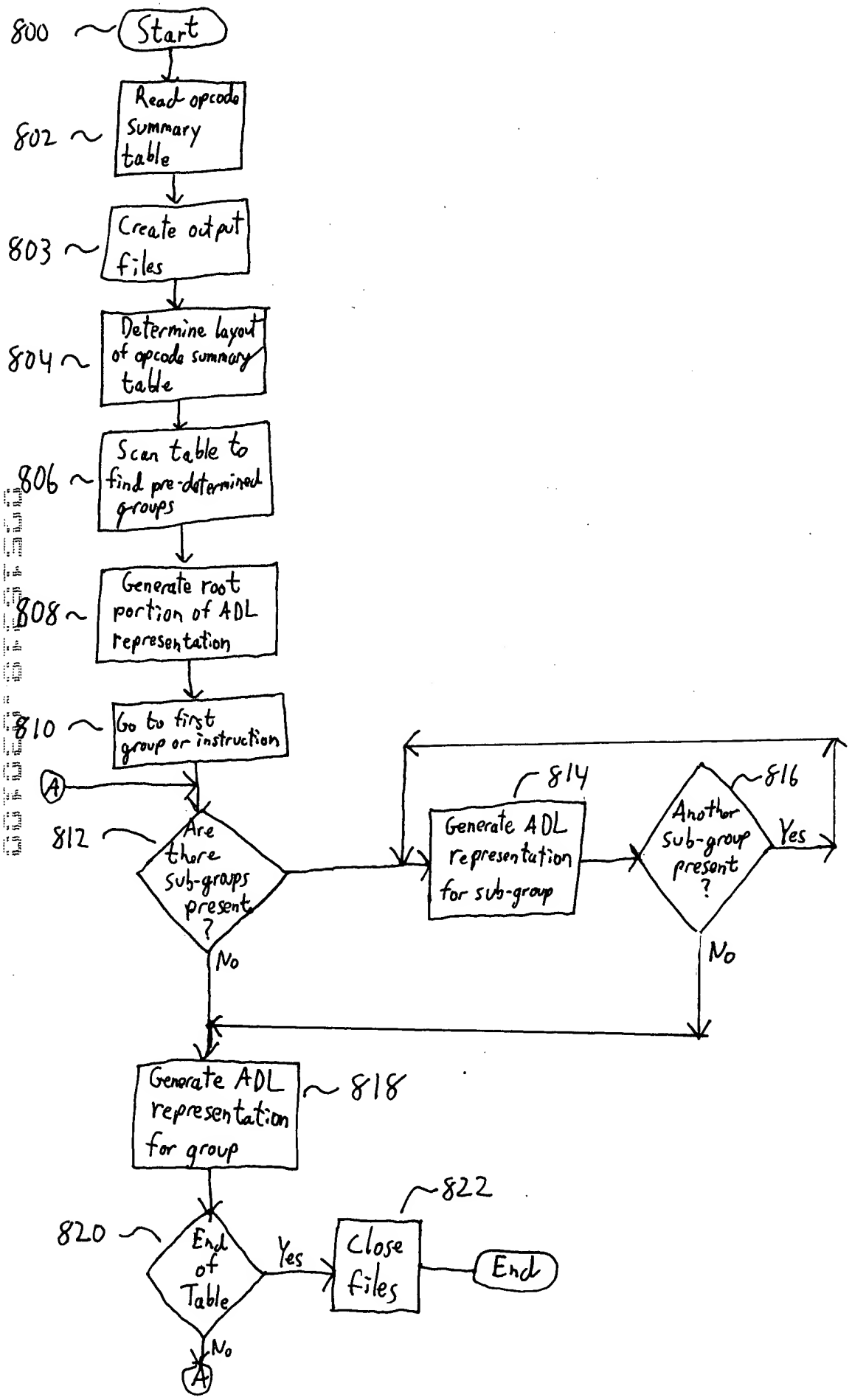
700

Arm Instruction Set

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Instruction	Description	Opnd Order
cond				0	0	1	*	*	*	*	S	Rn				Rd				rotate_imme				8_bit_immediate								*Data_proc_imm*	*Data_proc_imm*	cond
cond				0	0	1	1	1	0	1	S	0	0	0	0	Rd				rotate_imme				8_bit_immediate								MOVI		S
cond				0	0	1	1	1	1	1	S	0	0	0	0	Rd				rotate_imme				8_bit_immediate								MVNI		cp_num
cond				0	0	1	0	1	0	0	S	Rn				Rd				rotate_imme				8_bit_immediate								ADDI		CRd
cond				0	0	1	0	1	0	1	S	Rn				Rd				rotate_imme				8_bit_immediate								ADCI		opcode_1 CDP
cond				0	0	1	0	0	1	0	S	Rn				Rd				rotate_imme				8_bit_immediate								SUBI		opcode_1
cond				0	0	1	0	1	1	0	S	Rn				Rd				rotate_imme				8_bit_immediate								SBCI		Rd
cond				0	0	1	0	0	1	1	S	Rn				Rd				rotate_imme				8_bit_immediate								RSBI		CRn
cond				0	0	1	0	1	1	1	S	Rn				Rd				rotate_imme				8_bit_immediate								RSCI		CRm
cond				0	0	1	0	0	0	0	S	Rn				Rd				rotate_imme				8_bit_immediate								ANDI		opcode_2
cond				0	0	1	0	0	0	1	S	Rn				Rd				rotate_imme				8_bit_immediate								EORI		Rn
cond				0	0	1	1	1	0	0	S	Rn				Rd				rotate_imme				8_bit_immediate								ORRI		12_bit_offset
cond				0	0	1	1	1	1	0	S	Rn				Rd				rotate_imme				8_bit_immediate								BICI		8-bit_offset
cond				0	0	1	1	0	1	0	S	Rn				0	0	0	0	rotate_imme				8_bit_immediate								CMPI		8_bit_immediate
cond				0	0	1	1	0	1	1	S	Rn				0	0	0	0	rotate_imme				8_bit_immediate								CMNI		rotate_Immed
cond				0	0	1	1	0	0	0	S	Rn				0	0	0	0	rotate_imme				8_bit_immediate								TSTI		RdLo
cond				0	0	1	1	0	0	1	S	Rn				0	0	0	0	rotate_imme				8_bit_immediate								TEQI		RdHi
cond				0	0	0	*	*	*	*	S	Rn				Rd				shift_immed				shift		0					Rm	*Data_proc_imm*	*Data Processing*	Rm
cond				0	0	0	1	1	0	1	S	0	0	0	0	Rd				shift_immed				shift		0					Rm	MOVS		shift
cond				0	0	0	1	1	1	1	S	0	0	0	0	Rd				shift_immed				shift		0					Rm	MVNS		shift_Immed
cond				0	0	0	0	1	0	0	S	Rn				Rd				shift_immed				shift		0					Rm	ADDS		Rs
cond				0	0	0	0	1	0	1	S	Rn				Rd				shift_immed				shift		0					Rm	ADCS		register_list
cond				0	0	0	0	0	1	0	S	Rn				Rd				shift_immed				shift		0					Rm	SUBS		register_list_15
cond				0	0	0	0	1	1	0	S	Rn				Rd				shift_immed				shift		0					Rm	SBCS		24_bit_immedia
cond				0	0	0	0	0	1	1	S	Rn				Rd				shift_immed				shift		0					Rm	RSBS		
cond				0	0	0	0	1	1	1	S	Rn				Rd				shift_immed				shift		0					Rm	RSCS		
cond				0	0	0	0	0	0	0	S	Rn				Rd				shift_immed				shift		0					Rm	ANDS		
cond				0	0	0	0	0	0	1	S	Rn				Rd				shift_immed				shift		0					Rm	EORS		
cond				0	0	0	1	1	0	0	S	Rn				Rd				shift_immed				shift		0					Rm	ORRS		
cond				0	0	0	1	1	1	0	S	Rn				Rd				shift_immed				shift		0					Rm	BICS		
cond				0	0	0	1	0	1	0	S	Rn				0	0	0	0	shift_immed				shift		0					Rm	CMPS		
cond				0	0	0	1	0	1	1	S	Rn				0	0	0	0	shift_immed				shift		0					Rm	CMNS		
cond				0	0	0	1	0	0	0	S	Rn				0	0	0	0	shift_immed				shift		0					Rm	TSTS		
cond				0	0	0	1	0	0	1	S	Rn				0	0	0	0	shift_immed				shift		0					Rm	TEQS		
cond				0	0	0	*	*	*	*	S	Rn				Rd				Rs		0		shift		1					Rm	*Data_proc_reg*	*Data_proc_register_shift*	
cond				0	0	0	1	1	0	1	S	0	0	0	0	Rd				Rs		0		shift		1					Rm	MOVR		
cond				0	0	0	1	1	1	1	S	0	0	0	0	Rd				Rs		0		shift		1					Rm	MVNR		
cond				0	0	0	0	1	0	0	S	Rn				Rd				Rs		0		shift		1					Rm	ADDR		
cond				0	0	0	0	1	0	1	S	Rn				Rd				Rs		0		shift		1					Rm	ADCR		
cond				0	0	0	0	0	1	0	S	Rn				Rd				Rs		0		shift		1					Rm	SUBR		
cond				0	0	0	0	1	1	0	S	Rn				Rd				Rs		0		shift		1					Rm	SBCR		
cond				0	0	0	0	0	1	1	S	Rn				Rd				Rs		0		shift		1					Rm	RSBR		
cond				0	0	0	0	1	1	1	S	Rn				Rd				Rs		0		shift		1					Rm	RSCR		
cond				0	0	0	0	0	0	0	S	Rn				Rd				Rs		0		shift		1					Rm	ANDR		
cond				0	0	0	0	0	0	1	S	Rn				Rd				Rs		0		shift		1					Rm	EORR		
cond				0	0	0	1	1	0	0	S	Rn				Rd				Rs		0		shift		1					Rm	ORRR		
cond				0	0	0	1	1	1	0	S	Rn				Rd				Rs		0		shift		1					Rm	BICR		
cond				0	0	0	1	0	1	0	S	Rn				0	0	0	0	Rs		0		shift		1					Rm	CMPR		
cond				0	0	0	1	0	1	1	S	Rn				0	0	0	0	Rs		0		shift		1					Rm	CMNR		
cond				0	0	0	1	0	0	0	S	Rn				0	0	0	0	Rs		0		shift		1					Rm	TSTR		
cond				0	0	0	1	0	0	1	S	Rn				0	0	0	0	Rs		0		shift		1					Rm	TEQR		
cond				0	0	0	0	0	0	*	S	Rd				Rn				Rs		1	0	0	1						Rm	*Multiply*	*Multiply*	
cond				0	0	0	0	0	0	0	S	Rd				0	0	0	0	Rs		1	0	0	1						Rm	Mul	Multiply	
cond				0	0	0	0	0	0	1	S	Rd				Rn				Rs		1	0	0	1						Rm	MLA	Multiply Accumulate	
cond				0	0	0	0	1	*	*	S	RdHi				RdLo				Rs		1	0	0	1						Rm	*Multiply Long*	*Multiply Long*	
cond				0	0	0	0	1	1	1	S	RdHi				RdLo				Rs		1	0	0	1						Rm	SMLAL	Signed Multiply Accumulate Long	
cond				0	0	0	0	1	1	0	S	RdHi				RdLo				Rs		1	0	0	1						Rm	SMULL	Signed Multiply Long	
cond				0	0	0	0	1	0	1	S	RdHi				RdLo				Rs		1	0	0	1						Rm	UMLAL	Unsigned Multiply Accumulate Lo	
cond				0	0	0	0	1	0	0	S	RdHi				RdLo				Rs		1	0	0	1						Rm	UMULL	Unsigned Multiply Long	
cond				0	0	0	1	0	R	0	0	1	1	1	1	Rd				0	0	0	0	0	0	0	0	0	0	0	0	MRS	Move from Status register	
cond				0	0	1	1	0	R	1	0	field_mask				1	1	1	1	rotate_Immed				8_bit_immediate								MSRI	Move immediate to Status regidte	
cond				0	0	0	1	0	R	1	0	field_mask				1	1	1	1	0	0	0	0	0	0	0	0	Rm				MSR	Move register to Status regidter	
cond				1	0	1	L	24_bit_signed_offset																								B(L)	Brach instructio	Rd
cond				0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1		Rm	BX	Brach and exch	Rm
cond				0	1	0	*	*	*	*	*	Rn				Rd				12_bit_offset												*Load/Store imm*	*Load/Store imm*	Rs
cond				0	1	0	P	U	0	W	1	Rn				Rd				12_bit_offset												LDRI		Rn
cond				0	1	0	P	U	1	W	1	Rn				Rd				12_bit_offset												LDRBI		
cond				0	1	0	0	U	1	1	1	Rn				Rd				12_bit_offset												LDRBTI		
cond				0	1	0	0	U	0	1	1	Rn				Rd				12_bit_offset												LDRTI		
cond				0	1	0	P	U	0	W	0	Rn				Rd				12_bit_offset												STRI		
cond				0	1	0	P	U	1	W	0	Rn				Rd				12_bit_offset												STRBI		
cond				0	1	0	0	U	1	1	0	Rn				Rd				12_bit_offset												STRBTI		

706
702
710
712
714
716
704

Figure 8


800
Start
802
Read opcode summary table
803
Create output files
804
Determine layout of opcode summary table
806
Scan table to find pre-determined groups
808
Generate root portion of ADL representation
810
Go to first group or instruction
A
812
Are there sub-groups present ?
No
814
Generate ADL representation for sub-group
816
Another sub-group present ?
Yes
No
818
Generate ADL representation for group
820
End of Table
Yes
822
Close files
End
No
A